Abstract

This document provides general information about the JBoss EAP management CLI. It includes many examples that demonstrate how to use the CLI to administer and configure Red Hat JBoss Enterprise Application Platform. The Configuration Guide, Configuring Messaging, and other JBoss EAP documentation provide additional CLI command examples and detailed instructions for using the management CLI to accomplish specific administration tasks.
Specify the Host for Core Management and Runtime Commands 33
Specify the Server for Core Management and Runtime Commands 33

CHAPTER 6. CONFIGURING THE MANAGEMENT CLI  35
  6.1. PROPERTY SUBSTITUTION  37
  6.2. CREATING ALIASES  38
  6.3. .JBOSSCLIRC CONFIGURATION FILE  39
      Example .jbossclirc File  39
  6.4. USING VARIABLES  40
      Using the Set Command  40
      Using the Unset Command  40
      Using the jbossclirc File  40
      Using the Echo Command  40
      Example  41

CHAPTER 7. MANAGEMENT CLI COMMAND HISTORY  42
  View the Management CLI Command History  42
  Clear the Management CLI Command History  42
  Enable the Management CLI Command History  42
  Disable the Management CLI Command History  42

CHAPTER 8. MANAGEMENT CLI LOGGING  43
  Configure Management CLI Logging  43

CHAPTER 9. BATCH PROCESSING  44
  Batch Commands in External Files  44

CHAPTER 10. EMBEDDING A SERVER FOR OFFLINE CONFIGURATION  45
  Start an Embedded Standalone Server  45
      Specifying the Server Configuration  45
      Starting in Admin-only Mode  45
      Controlling Standard Output  46
      Boot Timeout  46
      Starting with a Blank Configuration  46
  Start an Embedded Host Controller  46
      Specifying the Host Controller Configuration  47
      Controlling Standard Output  47
      Boot Timeout  47
  Non-Modular Class Loading with the Management CLI  47

CHAPTER 11. HOW TO...  49
  11.1. ADD A DATASOURCE  49
  11.2. ADD AN EXTENSION  49
  11.3. ADD A JMS QUEUE  49
  11.4. ADD A JMS TOPIC  49
  11.5. ADD A MODULE  49
  11.6. ADD A SERVER  50
  11.7. ADD A SERVER GROUP  50
  11.8. ADD A SYSTEM PROPERTY  50
  11.9. CLONE A PROFILE  50
  11.10. CREATE A HIERARCHICAL PROFILE  50
  11.11. DEPLOY AN APPLICATION TO A MANAGED DOMAIN  50
  11.12. DEPLOY AN APPLICATION TO A STANDALONE SERVER  51
  11.13. DISABLE ALL APPLICATIONS  51
  11.14. DISPLAY THE ACTIVE USER  51
11.15. DISPLAY THE CONTENTS OF AN ATTACHMENT 51
11.16. DISPLAY SCHEMA INFORMATION 52
11.17. DISPLAY SYSTEM AND SERVER INFORMATION 52
11.18. ENABLE ALL DISABLED DEPLOYMENTS 53
11.19. GET THE COMMAND TIMEOUT VALUE 53
11.20. RELOAD A HOST CONTROLLER 53
11.21. RELOAD A HOST CONTROLLER IN ADMIN-ONLY MODE 53
11.22. RELOAD ALL SERVERS IN A SERVER GROUP 54
11.23. RELOAD A SERVER 54
11.24. RELOAD A STANDALONE SERVER 54
11.25. REMOVE AN EXTENSION 54
11.26. REMOVE A MODULE 54
11.27. RESET THE COMMAND TIMEOUT VALUE 55
11.28. RESTART ALL SERVERS IN A SERVER GROUP 55
11.29. RESTART A SERVER 55
11.30. SAVE THE CONTENTS OF AN ATTACHMENT 56
11.31. SET THE COMMAND TIMEOUT VALUE 56
11.32. SHUT DOWN A HOST CONTROLLER 56
11.33. SHUT DOWN THE SERVER 56
11.34. START ALL SERVERS IN A SERVER GROUP 57
11.35. START A SERVER 57
11.36. STOP ALL SERVERS IN A SERVER GROUP 57
11.37. STOP A SERVER 57
11.38. TAKE A CONFIGURATION SNAPSHOT 57
11.39. UNDEPLOY ALL APPLICATIONS 57
11.40. UNDEPLOY AN APPLICATION FROM A MANAGED DOMAIN 58
11.41. UNDEPLOY AN APPLICATION FROM A STANDALONE SERVER 58
11.42. UPDATE A HOST NAME 58
11.43. UPLOAD AN ATTACHMENT 58
11.44. VIEW A SERVER LOG 58

APPENDIX A. REFERENCE MATERIAL ................................................................. 59
A.1. MANAGEMENT CLI STARTUP ARGUMENTS 59
A.2. MANAGEMENT CLI BATCH MODE COMMANDS 60
A.3. MANAGEMENT CLI COMMANDS 61
A.4. MANAGEMENT CLI OPERATIONS 64
A.5. RESOURCE ATTRIBUTE DETAILS 67
CHAPTER 1. MANAGEMENT CLI OVERVIEW

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.
CHAPTER 2. GETTING STARTED WITH THE MANAGEMENT CLI

The management CLI is included with the JBoss EAP distribution. Once you launch the management CLI, you can connect to a running server instance or managed domains to perform management operations.

2.1. LAUNCH THE MANAGEMENT CLI

You can launch the management CLI by running the jboss-cli script provided with JBoss EAP.

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

**NOTE**

For Windows Server, use the EAP_HOME\bin\jboss-cli.bat script to launch the management CLI.

See Connect to the Server for details on launching the management CLI and connecting to the server in one step using the --connect argument.

**IMPORTANT**

The jboss-cli scripts set the com.ibm.jsse2.overrideDefaultTLS property to true. This setting is important if you are using the IBM JDK to prevent authentication issues when using SSL configured by Elytron.

Be sure to set this property if you are using the IBM JDK and using another method to start a CLI session, for example, programmatically using the classes available in EAP_HOME/bin/client/jboss-cli-client.jar.

For a complete listing of all available jboss-cli script arguments and their purposes, use the --help argument or see the Management CLI Startup Arguments section.

2.2. CONNECT TO THE SERVER

You can connect to a running standalone server or managed domain by using the connect command.

```
connect
```

The default host and port configuration is localhost:9990. If the server is listening on a different host and port, then these need to be provided to the connect command.

```
connect 192.168.0.1:9991
```

You can also launch the management CLI and connect to the server in one step using the --connect argument (and the --controller argument if necessary).

```
$ EAP_HOME/bin/jboss-cli.sh --connect --controller=192.168.0.1:9991
```

In JBoss EAP 7.3, to connect using the http-remoting protocol, use:
connect http-remoting://192.168.0.1:9990

2.3. GETTING HELP

The management CLI provides several ways for you to get help with using the management CLI.

- View general help on using the management CLI:
  
  help

  This provides detailed help with launching, navigating, and generating operation requests.

- View help for a particular command or operation:
  
  help COMMAND_OR_OPERATION

  This provides the usage, description, and arguments for the particular command or operation.

  For example:

  - View help for the patch command:
    
    help patch

  - View help for the apply action of the patch command:
    
    help patch apply

  - View help for the Elytron key-store resource add operation:
    
    help /subsystem=elytron/key-store=?:add

- View the list of available commands in the current context:
  
  help --commands

**NOTE**

Commands that require a connection to either a standalone server or domain controller do not appear in the list unless the connection has been established.

See the Management CLI Commands section for a listing of management CLI commands.

2.4. QUIT THE MANAGEMENT CLI

You can quit the management CLI by entering the quit command.

quit

2.5. RUN IN NON-INTERACTIVE MODE
You can issue management CLI commands without launching and interacting with the management CLI. This is useful for processing batches of commands and executing commands from scripts. You can either pass in the commands or pass in a file that contains commands to the `jboss-cli` startup script.

**Pass in Commands**
You can use the `--command` argument to provide a single CLI command to execute. The management CLI will terminate once the commands have completed.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --command="/interface=public:read-attribute(name=inet-address,resolve-expressions=true)"
```

The output from each command provided is displayed as it executes.

```
{
   "outcome" => "success",
   "result" => "127.0.0.1"
}
```

You can also use the `--commands` argument to provide a comma-separated list of CLI commands to execute.

**Pass in a File of Commands**
You can use the `--file` argument to pass in a text file of CLI commands to execute, with each command on a separate line in the file.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/cli_commands.txt
```

The output from each command in the file is displayed as it executes.

**Example Output**

```
{
   "outcome" => "success",
   "result" => "NORMAL"
}
helloworld.war
```

**NOTE**
You can include comments in a CLI script to aid understanding and maintenance. Comments are denoted by a pound sign (#) at the beginning of the line. While executing the script, JBoss EAP ignores the comments you included.

You can use the `--echo-command` argument to include the prompt and command with the output. This can be useful when resolving failures by matching the output to the command that was executed.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/cli_commands.txt --echo-command
```

The command and its output are displayed as it executes.

**Example Output With Command Echo**

```
[standalone@localhost:9990 /]:read-attribute(name=running-mode)
```
CHAPTER 2. GETTING STARTED WITH THE MANAGEMENT CLI

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

[standalone@localhost:9990 /] ls /deployment
helloworld.war
CHAPTER 3. NAVIGATING THE MANAGEMENT CLI

Many common terminal commands are available in the management CLI, such as `ls` to list the contents of a node path, `cd` to change the node path, and `pwd` to print the full node path. The management CLI also supports keyboard shortcuts.

3.1. CHANGE THE CURRENT PATH

You can change to a different node path by using the `cd` command and providing the desired path. When the management CLI is first launched, it is at the root level (`/`).

```
cd /subsystem=datasources
```
```
cd data-source=ExampleDS
```

3.2. PRINT THE CURRENT PATH

You can print the path of the current node by using the `pwd` command. When the management CLI is first launched, the path is at the root level (`/`).

```
cd /subsystem=undertow
```
```
cd server=default-server
```
```
pwd
```

The above example changes the path using the `cd` command and then outputs the following to the console:

```
/subsystem=undertow/server=default-server
```

3.3. LIST CONTENTS

You can list the contents of a particular node path by using the `ls` command. If the path ends on a node name, that resource’s attributes will be listed as well.

The below example navigates the `standard-sockets` socket binding group and then lists its contents.

```
cd /socket-binding-group=standard-sockets
```
```
ls -l
```

```
ATTRIBUTE         VALUE                                 TYPE
default-interface public                                STRING
name              standard-sockets                      STRING
port-offset       ${jboss.socket.binding.port-offset:0} INT
```

```
CHILD                                      MIN-OCCURS MAX-OCCURS
local-destination-outbound-socket-binding n/a    n/a
remote-destination-outbound-socket-binding n/a    n/a
socket-binding                             n/a    n/a
```

The same result can be achieved from anywhere in the resource tree hierarchy by specifying the node path to the `ls` command.
ls -l /socket-binding-group=standard-sockets

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>VALUE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-interface</td>
<td>public</td>
<td>STRING</td>
</tr>
<tr>
<td>name</td>
<td>standard-sockets</td>
<td>STRING</td>
</tr>
<tr>
<td>port-offset</td>
<td>${jboss.socket.binding.port-offset:0}</td>
<td>INT</td>
</tr>
</tbody>
</table>

CHILD MIN-OCCURS MAX-OCCURS

| local-destination-outbound-socket-binding | n/a | n/a |
| remote-destination-outbound-socket-binding | n/a | n/a |
| socket-binding                        | n/a | n/a |

You can also use the `--resolve-expressions` parameter to resolve the expressions of the returned attributes to their value on the server.

ls -l /socket-binding-group=standard-sockets --resolve-expressions

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>VALUE</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-interface</td>
<td>public</td>
<td>STRING</td>
</tr>
<tr>
<td>name</td>
<td>standard-sockets</td>
<td>STRING</td>
</tr>
<tr>
<td>port-offset</td>
<td>0</td>
<td>INT</td>
</tr>
</tbody>
</table>

CHILD MIN-OCCURS MAX-OCCURS

| local-destination-outbound-socket-binding | n/a | n/a |
| remote-destination-outbound-socket-binding | n/a | n/a |
| socket-binding                            | n/a | n/a |

In this example, the `port-offset` attribute shows its resolved value (0) instead of the expression (${jboss.socket.binding.port-offset:0}).

### 3.4. VIEW MULTI-PAGE OUTPUT

When you run the management CLI in interactive mode and the operation results in multiple pages of output, the command processor pauses the screen at the end of the first page. This allows you to page through the output one line or page at a time. The occurrence of multiple pages of output is indicated by a line of text displaying `--More(NNN)-->` at the end of the output.

The following is an example of a management CLI command that provides more than one page of output.

```json
/subsystem=undertow:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "default-security-domain" => "other",
    "default-server" => "default-server",
    "default-servlet-container" => "default",
    "default-virtual-host" => "default-host",
    "instance-id" => expression "${jboss.node.name}",
    "statistics-enabled" => false,
    "application-security-domain" => {"other" => {
      "enable-jacc" => false,
      "http-authentication-factory" => "application-http-authentication",
    },
```
Navigating Multi-page Output
When you encounter the line of text indicating that there is more output, you can proceed using one of the following options.

- Press **Enter** or the down arrow to page through the output one line at a time.
- Press the Spacebar or **PgDn** to skip to the next page of output.
- Press **PgUp** to return to the previous page of output.
- Press **Home** to return to the beginning of the output.
- Press **End** to skip to the final line of the output.
- Type **q** to interrupt the command and exit.

**NOTE**
On Windows the **PgUp**, **PgDn**, **Home**, and **End** keys are available beginning with Windows Server 2016. There are no issues with other operating systems.

Searching Multi-page Output
You can search for text within multi-page output.

1. Use a forward slash (/) to initiate searching.
2. Type the desired text and press **Enter** to search.
   - Press **n** to go to the next match.
   - Press **N** to go to the previous match.

You can also use the up and down arrows to browse through the search history.

### 3.5. USE KEYBOARD NAVIGATION SHORTCUTS

When running the management CLI in interactive mode, you can use keyboard shortcuts to quickly edit a management CLI command.

**NOTE**
You can also use the Tab key to autocomplete a portion of a management CLI command or view the available options.

The keyboard shortcuts you can use vary depending on which supported platform you are running:

- **Red Hat Enterprise Linux**
- Windows Server
- Solaris

### Table 3.1. Red Hat Enterprise Linux Keyboard Navigation Shortcuts

<table>
<thead>
<tr>
<th>Navigation</th>
<th>Keyboard Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left one word</td>
<td>Alt+B or Ctrl+left arrow</td>
</tr>
<tr>
<td>Right one word</td>
<td>Alt+F or Ctrl+right arrow</td>
</tr>
<tr>
<td>Beginning of the line</td>
<td>Ctrl+A or Home</td>
</tr>
<tr>
<td>End of the line</td>
<td>Ctrl+E or End</td>
</tr>
<tr>
<td>Left one character</td>
<td>Ctrl+B or left arrow</td>
</tr>
<tr>
<td>Right one character</td>
<td>Ctrl+F or left arrow</td>
</tr>
</tbody>
</table>

### Table 3.2. Windows Server Keyboard Navigation Shortcuts

<table>
<thead>
<tr>
<th>Navigation</th>
<th>Keyboard Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left one word</td>
<td>Alt+B</td>
</tr>
<tr>
<td>Right one word</td>
<td>Alt+F</td>
</tr>
<tr>
<td>Beginning of the line</td>
<td>Ctrl+A or Home</td>
</tr>
<tr>
<td>End of the line</td>
<td>Ctrl+E or End</td>
</tr>
<tr>
<td>Left one character</td>
<td>Ctrl+B or left arrow</td>
</tr>
<tr>
<td>Right one character</td>
<td>Ctrl+F or left arrow</td>
</tr>
</tbody>
</table>

### Table 3.3. Solaris Keyboard Navigation Shortcuts

<table>
<thead>
<tr>
<th>Navigation</th>
<th>Keyboard Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left one word</td>
<td>Alt+B or Ctrl+left arrow</td>
</tr>
<tr>
<td>Right one word</td>
<td>Alt+F or Ctrl+right arrow</td>
</tr>
<tr>
<td>Beginning of the line</td>
<td>Ctrl+A or Home</td>
</tr>
<tr>
<td>End of the line</td>
<td>Ctrl+E or End</td>
</tr>
<tr>
<td>Navigation</td>
<td>Keyboard Shortcut</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Left one character</td>
<td>Ctrl+B or left arrow</td>
</tr>
<tr>
<td>Right one character</td>
<td>Ctrl+F or left arrow</td>
</tr>
</tbody>
</table>
CHAPTER 4. CREATING AND EXECUTING REQUESTS

JBoss EAP configuration is presented as a hierarchical tree of addressable resources, each offering their own set of operations. Management CLI operation requests allow for low-level interaction with the management model and provide a controlled way to edit server configurations.

Operation requests use the following format:

```
/NODE_TYPE=NODE_NAME:OPERATION_NAME(PARAMETER_NAME=PARAMETER_VALUE)
```

An operation request consists of three parts:

- **address**
  
  The address specifies the resource node on which to perform the operation. `NODE_TYPE` maps to an element name and `NODE_NAME` maps to that element’s `name` attribute in the configuration XML. Each level of the resource tree is separated by a slash (`/`).

- **operation name**

  The operation to perform on the resource node. It is prefixed with a colon (`:`).

- **parameters**

  The set of required or optional parameters that vary depending on the operation. They are contained within parentheses (`()`).

### Construct an Operation Request

1. **Determine the address**

   You can reference the XML configuration files (standalone.xml, domain.xml, or host.xml) to assist in determining the required address. You can also use tab completion to view the available resources.

   Below are several common addresses for resources at the root (`/`) level.

   - `/deployment=DEPLOYMENT_NAME` - Deployment configurations.
   - `/socket-binding-group=SOCKET_BINDING_GROUP_NAME` - Socket binding configurations.
   - `/interface=INTERFACE_NAME` - Interface configurations.
   - `/subsystem=SUBSYSTEM_NAME` - Subsystem configuration when running as a standalone server.
   - `/profile=PROFILE_NAME/subsystem=SUBSYSTEM_NAME` - Subsystem configuration for the selected profile when running in a managed domain.
   - `/host=HOST_NAME` - Server configuration for the selected host when running in a managed domain.

   The below address is for the ExampleDS datasource.

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

2. **Determine the operation**
The available operations differ for each type of resource node. You can use the :read-operation-names operation on a resource address to view the available operations. You can also use tab completion.

Use the :read-operation-description operation to get information on a particular operation for a resource.

The below operation (once the appropriate parameters are included) will set the value of an attribute for the ExampleDS datasource.

```
/subsystem=datasources/data-source=ExampleDS:write-attribute
```

3. Determine the parameters
Each operation has its own set of available parameters. If you attempt to perform an operation without a required parameter, you will receive an error message that the parameter cannot be null.

Multiple parameters are separated by commas (,). If an operation does not have any parameters, then the parentheses are optional.

Use the :read-operation-description operation on a resource, passing in the operation name, to determine the required parameters for that operation. You can also use tab completion to list the available parameters.

The below operation disables the ExampleDS datasource by setting its enabled attribute to false.

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

Once entered, the management interface will perform the operation request on the server configuration. Depending on the operation request, you will receive output to the terminal that contains the outcome and the result or response of the operation.

The following response from disabling the ExampleDS datasource shows that the operation was successful and requires a reload of the server in order to take effect.

```
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

You can use the read-attribute operation to read the value of the ExampleDS datasource’s enabled attribute.

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

The following response shows that the operation was successful and that the value of enabled is false.

```
{
    "outcome" => "success",
```
"result" => false,

4.1. DISPLAY RESOURCE VALUES

You can use the `read-resource` operation to view a resource’s attribute values.

You can specify parameters to provide complete information about child resources, recursively. You can also specify parameters to include runtime attributes, resolve expressions, and include aliases. Use `read-operation-description(name=read-resource)` to see the description of all available parameters for `read-resource`.

The following example reads the attributes for a deployment. It includes details such as the deployment name, whether it is enabled, and the last time it was enabled.

```
/deployment=DEPLOYMENT_NAME:read-resource
{
  "outcome" => "success",
  "result" => {
    "enabled" => true,
    "enabled-time" => 1453929902598L,
    "enabled-timestamp" => "2016-01-27 16:25:02,598 EST",
    "name" => "DEPLOYMENT_NAME",
    "owner" => undefined,
    "persistent" => true,
    "runtime-name" => "DEPLOYMENT_NAME",
    "subdeployment" => undefined,
    "subsystem" => {
      "undertow" => undefined,
      "logging" => undefined
    }
  }
}
```

Include Runtime Attributes

The `include-runtime` parameter can be used to retrieve runtime attributes.

The following example reads the attributes for a deployment. In addition to persistent attributes, it also includes runtime attributes, such as the deployment status and the last time it was disabled.

```
/deployment=DEPLOYMENT_NAME:read-resource(include-runtime=true)
{
  "outcome" => "success",
  "result" => {
    "disabled-time" => undefined,
    "disabled-timestamp" => undefined,
    "enabled" => true,
    "enabled-time" => 1453929902598L,
    "enabled-timestamp" => "2016-01-27 16:25:02,598 EST",
    "name" => "DEPLOYMENT_NAME",
```
"owner" => undefined,
"persistent" => true,
"runtime-name" => "DEPLOYMENT_NAME",
"status" => "OK",
"subdeployment" => undefined,
"subsystem" => {
  "undertow" => undefined,
  "logging" => undefined
}
}
}

You can also use the not operator (!) when passing in boolean parameters. For example:

- :read-resource(include-runtime=false) can be entered as :read-resource(!include-runtime)
- :read-resource(include-runtime=true) can be entered as :read-resource(include-runtime)

Read Child Resources Recursively
The recursive parameter can be used to retrieve attributes recursively from child resources.

The following example reads the attributes for a deployment. In addition to the resource’s own attributes, it recursively returns the attributes for its child resources, such as the undertow subsystem configuration.

/deployment=DEPLOYMENT_NAME:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    ...
    "enabled" => true,
    "enabled-time" => 1453929902598L,
    "enabled-timestamp" => "2016-01-27 16:25:02,598 EST",
    "name" => "DEPLOYMENT_NAME",
    "owner" => undefined,
    "persistent" => true,
    "runtime-name" => "DEPLOYMENT_NAME",
    "subdeployment" => undefined,
    "subsystem" => {
      "undertow" => {
        "context-root" => "/test",
        "server" => "default-server",
        "virtual-host" => "default-host",
        "servlet" => undefined,
        "websocket" => undefined
      },
      "logging" => {"configuration" => undefined}
    }
  }
}

Exclude Default Values
The include-defaults parameter can be used to show or hide default values when reading attributes for a resource. This is true by default, meaning that default values will be shown when using the read-resource operation.
The following example uses the `read-resource` operation on the `undertow` subsystem.

```
/subsystem=undertow:read-resource
{
  "outcome" => "success",
  "result" => {
    "default-security-domain" => "other",
    "default-server" => "default-server",
    "default-servlet-container" => "default",
    "default-virtual-host" => "default-host",
    "instance-id" => expression "$\{jboss.node.name\}\",
    "statistics-enabled" => false,
    "buffer-cache" => {"default" => undefined},
    "configuration" => {
      "filter" => undefined,
      "handler" => undefined
    },
    "server" => {"default-server" => undefined},
    "servlet-container" => {"default" => undefined}
  }
}
```

The following example also uses the `read-resource` operation on the `undertow` subsystem, but sets the `include-defaults` parameter to `false`. Several attributes, such as `statistics-enabled` and `default-server`, now display `undefined` as their value instead of the default value.

```
/subsystem=undertow:read-resource(include-defaults=false)
{
  "outcome" => "success",
  "result" => {
    "default-security-domain" => undefined,
    "default-server" => undefined,
    "default-servlet-container" => undefined,
    "default-virtual-host" => undefined,
    "instance-id" => undefined,
    "statistics-enabled" => undefined,
    "buffer-cache" => {"default" => undefined},
    "configuration" => {
      "filter" => undefined,
      "handler" => undefined
    },
    "server" => {"default-server" => undefined},
    "servlet-container" => {"default" => undefined}
  }
}
```

**Resolve Expressions**

The `resolve-expressions` parameter can be used to resolve the expressions of the returned attributes to their value on the server.

Attributes with expressions as their values use the format `$\{PARAMETER:DEFAULT_VALUE\}`. See Property Replacement in the Configuration Guide for more information.

The following example reads the attributes for a deployment. The `instance-id` attribute shows its resolved value (`test-name`) instead of the expression (`$\{jboss.node.name\}`).
### 4.2. DISPLAY RESOURCE DESCRIPTIONS

You can use the `read-resource-description` operation to a description about a resource and its attributes.

```plaintext
:read-resource-description
```

You can specify parameters to provide complete descriptions about child resources, recursively. You can also specify parameters to include details of the resource’s operations and notifications. Use `read-operation-description(name=read-resource-description)` to see the description of all available parameters for `read-resource-description`.

The following example displays the attribute details for a buffer cache.

```plaintext
/subsystem=undertow/buffer-cache=default:read-resource-description
{
  "outcome" => "success",
  "result" => {
    "description" => "The buffer cache used to cache static content",
    "attributes" => {
      "buffer-size" => {
        "type" => INT,
        "description" => "The size of an individual buffer",
        "expressions-allowed" => true,
        "nillable" => true,
        "default" => 1024,
        "min" => 0L,
        "max" => 2147483647L,
        "access-type" => "read-write",
        "storage" => "configuration",
        "restart-required" => "resource-services"
      }
    }
  }
}
```
"description" => "The numbers of buffers in a region",
"expressions-allowed" => true,
"nillable" => true,
"default" => 1024,
"min" => 0L,
"max" => 2147483647L,
"access-type" => "read-write",
"storage" => "configuration",
"restart-required" => "resource-services"
},
"max-regions" => {
  "type" => INT,
  "description" => "The maximum number of regions",
  "expressions-allowed" => true,
  "nillable" => true,
  "default" => 10,
  "min" => 0L,
  "max" => 2147483647L,
  "access-type" => "read-write",
  "storage" => "configuration",
  "restart-required" => "resource-services"
}
},
"operations" => undefined,
"notifications" => undefined,
"children" => {}
}

See Resource Attribute Details to learn more about the fields returned for attributes.

### 4.3. DISPLAY AN ATTRIBUTE VALUE

You can use the **read-attribute** operation to view the current value of a single attribute.

:![](image)

The following example displays the log level for the root logger by reading the **level** attribute.

!/subsystem=logging/root-logger=ROOT:read-attribute(name=level)

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

One advantage of using the **read-attribute** operation is the ability to expose the current runtime value of an attribute.

!/interface=public:read-attribute(name=resolved-address)

```
{
  "outcome" => "success",
  "result" => "127.0.0.1"
}
```
The **resolved-address** attribute is a runtime attribute. This attribute is not displayed when using the **read-resource** operation on the public interface unless you pass in the **include-runtime** parameter. And even then, it is displayed with the rest of the resource’s other attributes.

You can also use the **include-defaults** and **resolve-expressions** parameters. See Display Resource Values for details on these parameters.

### 4.4. UPDATE AN ATTRIBUTE

You can use the **write-attribute** operation to update the value of an attribute for a resource.

```
write-attribute(name=ATTRIBUTE_NAME, value=ATTRIBUTE_VALUE)
```

The following example disables the deployment scanner by setting the **scan-enabled** attribute to **false**.

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

The response from the operation request shows that it was successful. You can also confirm the result by using the **read-attribute** operation to read the **scan-enabled** attribute, which now shows as **false**.

```
/subsystem=deployment-scanner/scanner=default:read-attribute(name=scan-enabled)
```

### 4.5. UNDEFINE AN ATTRIBUTE

You can set the value of an attribute to **undefined**. If this attribute has a default value, that will be used.

The following example undefines the **level** attribute for the root logger.

```
/subsystem=logging/root-logger=ROOT:undefine-attribute(name=level)
```

The default value for the **level** attribute is **ALL**. You can see that this default is used when performing the **read-resource** operation.

```
/subsystem=logging/root-logger=ROOT:read-resource
```

To view the resource without the default values being read, you must use the **include-defaults** parameter.
To view the resource without the default values being read, you must use set the `include-defaults` parameter to **false**. You can now see that the value of `level` is **undefined**.

```
/subsystem=logging/root-logger=ROOT:read-resource(include-defaults=false)
{
    "outcome" => "success",
    "result" => {
        "filter" => undefined,
        "filter-spec" => undefined,
        "handlers" => [  
            "CONSOLE",
            "FILE"
        ],
        "level" => undefined
    }
}
```

### 4.6. DISPLAY OPERATION NAMES

You can use the `read-operation-names` list the available operations for a given resource.

```
:read-operation-names
```

The following example lists the available operations to perform on a deployment.

```
/deployment=DEPLOYMENT_NAME:read-operation-names
{
    "outcome" => "success",
    "result" => [
        "add",
        "deploy",
        "list-add",
        "list-clear",
        "list-get",
        "list-remove",
        "map-clear",
        "map-get",
        "map-put",
        "map-remove",
        "query",
        "read-attribute",
        "read-attribute-group",
        "read-attribute-group-names",
        "read-children-names",
        "read-children-resources",
        "read-children-types",
        "read-operation-description",
        "read-operation-names",
        "read-resource",
        "read-resource-description",
        "redeploy",
        "remove",
        "undefine-attribute",
        "undeploy",
    ]
}
Use the `read-operation-description` operation to display an operation description.

4.7. DISPLAY AN OPERATION DESCRIPTION

You can use the `read-operation-description` operation to display a description of a certain operation for a resource. This also includes parameter descriptions and which parameters are required.

```
:read-operation-description(name=OPERATION_NAME)
```

The following example provides the description and parameter information for the `add` operation on a system property.

```
/system-property=SYSTEM_PROPERTY:read-operation-description(name=add)
{
  "outcome" => "success",
  "result" => {
    "operation-name" => "add",
    "description" => "Adds a system property or updates an existing one.",
    "request-properties" => {
      "value" => {
        "type" => STRING,
        "description" => "The value of the system property.",
        "expressions-allowed" => true,
        "required" => false,
        "nillable" => true,
        "min-length" => 0L,
        "max-length" => 2147483647L
      }
    },
    "reply-properties" => {},
    "read-only" => false,
    "runtime-only" => false
  }
}
```

4.8. ADD A VALUE WITH SPECIAL CHARACTERS

Occasionally when creating management CLI requests, you may need to add a value that contains special characters. Certain special characters, such as those used in the syntax of a management CLI request, must be entered in a particular manner.

In many cases, but not all, enclosing the value in double quotes (""") is sufficient. If you are unsure whether your special character was accepted properly, be sure to read the attribute or resource after adding the value to verify that it was saved correctly.

See the sections below for information on how to process the following special characters.

- **Whitespace**
- **Quotation Marks**
Whitespace
By default, whitespace is trimmed from values added through the management CLI. You can include a space in a value by enclosing the value in double quotes ("""") or braces ({}), or by escaping it using a backslash (\).

```
/system-property=test1:add(value="Hello World")
/system-property=test2:add(value={Hello World})
/system-property=test3:add(value=Hello\ World)
```

This will set the value to **Hello World**.

Quotation Marks
You can use a single quotation mark (') in a value by enclosing the value in double quotes ("""") or by escaping it using a backslash (\). The following examples set the value of system properties to **server's**.

```
/system-property=test1:add(value="server's")
/system-property=test2:add(value=server\'s)
```

You can use a double quotation mark (") in a value by escaping it using a backslash (\). Depending on the location of the quotation mark in the value, you might also need to enclose the value in double quotes (""""). The following example sets the value of a system property to **"quote"**.

```
/system-property=test1:add(value="\"quote\""")
```

Commas
You can use a comma (,) in a value by enclosing the value in double quotes ("""").

```
/system-property=test:add(value="Last,First")
```

This will set the value to **Last,First**.

Parentheses
You can include parentheses (()) in a value by enclosing the value in double quotes ("""") or braces ({}), or by escaping the parenthesis using a backslash (\).

```
/system-property=test1:add(value="one(1)")
/system-property=test2:add(value={one(1)})
/system-property=test3:add(value=one\(1\))
```

This will set the value to **one(1)**.

Braces
You can include braces ({} in a value by enclosing the value in double quotes ("""").

```
```
This will set the value to `{braces}`.

**Brackets**
You can include brackets ([ ]) in a value by enclosing the value in double quotes ( "" ).

This will set the value to `[brackets]`.

**Diacritic Marks**
Diacritic marks, such as ñ, ř, or ý, can be used when adding a value using the management CLI.

However, do not enclose the value in double quotes ("""). This can cause the diacritic mark to be replaced with a question mark (?). If the value has whitespace that needs to be maintained, instead enclose the value in braces ({}) or escape the space with a backslash (\).

This will set the value to `Dos años`.

### 4.9. SPECIFY OPERATION HEADERS

You can specify operation headers to control certain aspects of how an operation executes. The following operation headers are available:

**allow-resource-service-restart**

Whether to restart runtime services that require a restart in order for the operation’s changes take effect. This defaults to `false`.

**WARNING**

Using the `allow-resource-service-restart=true` header can disrupt end-user request handling until the required services have been restarted.

**blocking-timeout**

The maximum time in seconds that the operation should block at any one point in its completion process before the operation is rolled back. This defaults to 300 seconds.

**roles**

The list of RBAC roles that should be used when making access control decisions instead of those from the roles normally associated with the user invoking the operation. Note that this can only be used to reduce permissions for a caller, not to increase permissions.
rollback-on-runtime-failure

Whether persistent configuration changes should be reverted if applying the changes to runtime services fails. This defaults to true.

rollout

The rollout plan for a managed domain deployment. For more information, see the Using Rollout Plans section of the JBoss EAP Configuration Guide.

Example: Deploy an Application Using an Operation Header

```
deployment deploy-file /path/to/DEPLOYMENT.war --headers={allow-resource-service-restart=true}
```

Example: Remove a Resource Using an Operation Header

```
/subsystem=infinispan/cache-container=test:remove() {allow-resource-service-restart=true}
```

Use a semicolon (;) to separate multiple operation headers.

4.10. USE IF-ELSE CONTROL FLOW

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

**NOTE**

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

Expressions can contain any of the following items:

- Parentheses to group and prioritize expressions
- Conditional operators
  - and (&&)
  - or (||)
- Comparison operators
  - equal to (==)
  - not equal to (!=)
  - greater than (>)
  - greater than or equal to (>=)
  - less than (<)
  - less than or equal to (<=)
  - match regular expression (~=)
IMPORTANT

The match regular expression (~=) operator is provided as Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

The below example uses the match regular expression (~=) operator to check whether the value of the features system property contains jgroups.

```
if (result ~= ".*jgroups.*") of /:resolve-expression(expression=${features})
  echo Configuring JGroups
end-if
```

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

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

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

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

The below example issues the appropriate management CLI command to enable the ExampleDS datasource by checking the launch type of the server process (STANDALONE or DOMAIN).

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

Management CLI commands with if-else control flow can be specified in a file, with each command on a separate line in the file. You can then pass the file to the jboss-cli script to be executed non-interactively using the --file parameter.

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=CLI_FILE
```
4.11. USE TRY-CATCH-FINALLY CONTROL FLOW

The management CLI provides a simplified try-catch-finally control flow. It consists of three sets of operations and commands corresponding to the try, catch, and finally blocks. The catch and finally blocks are optional, but at least one of them should be present and only one catch block can be specified.

The control flow begins with execution of the try batch. If the try batch completes successfully, then the catch batch is skipped and the finally batch is executed. If the try batch fails, for example, java.io.IOException, the try-catch-finally control flow will terminate immediately, and the catch batch if it is available will be executed. The finally batch will always execute at the end of the control flow whether the try and catch batches succeeds or fails to execute.

There are four commands that define the try-catch-finally control flow:

- **try** command starts the try batch. The try batch continues until one of catch or finally command is encountered.
- **catch** command marks the end of the try batch. The try batch is then held back and the catch batch is started.
- **finally** command marks the end of the catch batch or the try batch and starts the finally batch.
- **end-try** is the command that ends either the catch or finally batch and runs the try-catch-finally control flow.

The following example either creates or re-creates a datasource and enables it:

```
try
/subsystem=datasources/data-source=myds:add(connection-url=CONNECTION_URL,jndi-name=java:/myds,driver-name=h2)
catch
/subsystem=datasources/data-source=myds:remove
/subsystem=datasources/data-source=myds:add(connection-url=CONNECTION_URL,jndi-name=java:/myds,driver-name=h2)
finally
/subsystem=datasources/data-source=myds:enable
end-try
```

4.12. USE FOR-DONE CONTROL FLOW

The management CLI supports for-done control flow, which allows you to iterate over a collection returned from an operation and execute commands on each item in the collection. You can use for-done statements with the management CLI in interactive or non-interactive mode. for-done statements use the following syntax:

```
for VARIABLE_NAME in OPERATION
   COMMANDS_TO_EXECUTE
done
```

- The VARIABLE_NAME can be used in the COMMANDS_TO_EXECUTE using the syntax $VARIABLE_NAME.
The **OPERATION** must return a collection.

- The **COMMANDS_TO_EXECUTE** is a list of commands to execute, with each command on a separate line.

The following example iterates over all deployments and displays whether each is enabled or disabled.

```bash
for deploymentName in :read-children-names(child-type=deployment)
  if (result == true) of /deployment=$deploymentName:read-attribute(name=enabled)
    echo $deploymentName is enabled.
  else
    echo $deploymentName is disabled.
  end-if
done
```

Enter `done --discard` if you want to discard the current **for** block without executing the commands.

**NOTE**

The use of **for-done** statements while in batch mode, or nesting them inside other **for-done** statements, is not supported.

### 4.13. QUERY A RESOURCE

The JBoss EAP management CLI provides the **query** operation to query a resource. You can use the **:read-resource** operation to read all attributes for a resource. If you want to list only selected attributes, you can use the **:query** operation.

For example, to see the list of **name** and **enabled** attributes, use the following command:

```
/deployment=jboss-modules.jar:query(select=["name","enabled"])
```

The following response shows that the operation was successful. The **name** and **enabled** attributes are listed for the **jboss-modules.jar** deployment.

```
{
  "outcome" => "success",
  "result" => {
    "name" => "jboss-modules.jar",
    "enabled" => true
  }
}
```

You can also query across multiple resources, for example, list the **name** and **enabled** attributes of all deployments, using a wildcard:

```
/deployment=*:query(select=["name","enabled"])
```

The following response shows that the operation was successful. The **name** and **enabled** attributes of all deployments are listed.

```
{
  "outcome" => "success",
```
"result" => [  
    {  
      "address" => ["deployment" => "helloworld.war"],  
      "outcome" => "success",  
      "result" => {  
        "name" => "helloworld.war",  
        "enabled" => true  
      }  
    },  
    {  
      "address" => ["deployment" => "kitchensink.war"],  
      "outcome" => "success",  
      "result" => {  
        "name" => "kitchensink.war",  
        "enabled" => true  
      }  
    },  
    {  
      "address" => ["deployment" => "xyz.jar"],  
      "outcome" => "success",  
      "result" => {  
        "name" => "xyz.jar",  
        "enabled" => false  
      }  
    }  
  ]}

The :query operation also filters relevant objects. For example, to view the name and enabled attribute values for deployments with enabled as true.

```
/deployment=*/:query(select=["name","enabled"],where=["enabled","true"])```

The following response shows that the operation was successful. The name and enabled attribute values for deployments with enabled as true are listed.

```
"outcome" => "success",  
"result" => [  
    {  
      "address" => ["deployment" => "helloworld.war"],  
      "outcome" => "success",  
      "result" => {  
        "name" => "helloworld.war",  
        "enabled" => true  
      }  
    },  
    {  
      "address" => ["deployment" => "kitchensink.war"],  
      "outcome" => "success",  
      "result" => {  
        "name" => "kitchensink.war",  
        "enabled" => true  
      }  
    }  
  ]
```
4.14. REDIRECT OUTPUT

Instead of printing output from a management CLI operation to the terminal, you can redirect the output to a file or to another command.

Redirect Output to a File

Use the `>` operator to redirect output from a management CLI operation to a file on the file system.

**Example: Write read-resource Output to a File**

```
:read-resource > myfile.txt
```

Use the `>>` operator to redirect output from a management CLI operation and append it to a file on the file system.

**Example: Append read-resource Output to a File**

```
:read-resource >> myfile.txt
```

Redirect Output to a Command

Use the `|` operator to redirect output from a management CLI operation to the `grep` command to search output for a regular expression match. Currently, `grep` is the only supported command for the `|` operator.

**Example: Search Output from the server.log File**

```
/subsystem=logging/log-file=server.log:read-log-file | grep Deployed
```

NOTE

It is not supported to use the `grep` command more than once in the same command.
CHAPTER 5. USING THE MANAGEMENT CLI WITH A MANAGED DOMAIN

You can use the management CLI to configure and manage both standalone servers and managed domains. The JBoss EAP documentation usually shows examples of management CLI commands for a standalone server configuration. If you are running a managed domain instead, you often need to adjust the command. The following sections describe how to change standalone server management CLI commands for a managed domain configuration.

Specify the Profile for Subsystem Configuration

The management CLI commands for standalone server subsystem configuration begin with `/subsystem=SUBSYSTEM_NAME`. For managed domain subsystem configuration, you must specify which profile's subsystem to configure by starting the command with `/profile=PROFILE_NAME/subsystem=SUBSYSTEM_NAME`.

Example: Read the Logging Subsystem Configuration (Standalone Server)

```
/subsystem=logging:read-resource
```

This example shows how to read the configuration of the `logging` subsystem for a standalone server.

Example: Read the Logging Subsystem Configuration (Managed Domain)

```
/profile=default/subsystem=logging:read-resource
```

This example shows how to read the configuration of the `logging` subsystem for a the `default` profile in a managed domain.

Specify the Host for Core Management and Runtime Commands

Some core management and runtime commands for a managed domain require you to specify the host that the command applies to by starting the command with `/host=HOST_NAME`.

Example: Enable Audit Logging (Standalone Server)

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

This example shows how to enable audit logging for a standalone server.

Example: Enable Audit Logging (Managed Domain)

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

This example shows how to enable audit logging for the `master` host in a managed domain.

**NOTE**

Some commands require the host as an argument, for example, `reload --host=HOST_NAME`. If you do not specify a host for these commands, an error message notifies you that the `--host` argument is required.

Specify the Server for Core Management and Runtime Commands
Some core management and runtime commands for a managed domain require you to specify the host and server that the command applies to by starting the command with `/host=HOST_NAME/server=SERVER_NAME`.

**Example: Display Runtime Metrics for a Deployment (Standalone Server)**

```
/deployment=test-application.war/subsystem=undertow:read-attribute(name=active-sessions)
```

This example shows how to display runtime metrics for a standalone server deployment.

**Example: Display Runtime Metrics for a Deployment (Managed Domain)**

```
/host=master/server=server-one/deployment=test-application.war/subsystem=undertow:read-attribute(name=active-sessions)
```

This example shows how to display runtime metrics for a managed domain deployment that is deployed to the server-one server on the master host.
CHAPTER 6. CONFIGURING THE MANAGEMENT CLI

Certain aspects of the management CLI can be customized in its configuration file, `jboss-cli.xml`. This file must be located either in the `EAP_HOME/bin` directory or in a custom directory specified with the `jboss.cli.config` system property.

The following elements can be configured in the `jboss-cli.xml` file.

**default-protocol**

The default protocol to use when controller addresses are supplied without one. The default is `remote+http`. If port `9990` is used and no protocol is specified, the protocol will automatically default to `remoting` unless you set the `use-legacy-override` attribute to `false`.

**default-controller**

Configuration of the controller to which to connect if the `connect` command is executed without any parameters. If the management CLI is started with the argument `--controller=` or `controller=`, then the value specified in the argument overrides the `default-controller` definition from the configuration.

- **protocol** - Protocol name of the controller. If one is not provided, the value of `default-protocol` will be used.
- **host** - Host name of the controller. The default is `localhost`.
- **port** - Port number on which to connect to the controller. The default value is `9990`.

**controllers**

You can define connection controller aliases in the `jboss-cli.xml` file. For example:

```xml
<!-- The default controller to connect to when 'connect' command is executed w/o arguments -->
<default-controller>
  <host>localhost</host>
  <port>9990</port>
</default-controller>

<!-- CLI connection controller aliases -->
<controllers>
  <controller name="ServerOne">
    <protocol>remoting</protocol>
    <host>192.168.3.45</host>
    <port>9990</port>
  </controller>
  <controller name="ServerTwo">
    <protocol>http-remoting</protocol>
    <host>192.168.3.46</host>
  </controller>
</controllers>
```

The `name` attribute of controller element should be used as a value to `--controller=` argument. For example, `--controller=ServerTwo`.

**validate-operation-requests**

Whether to validate the parameter list of operation requests before sending the requests to the controller for execution. The default is `true`.

**history**
The configuration for the CLI command history log.

- **enabled** - Whether or not the **history** is enabled. The default is **true**.
- **file-name** - The file name in which the history will be stored. The default is `.jboss-cli-history`.
- **file-dir** - The directory in which the history is to be stored. The default is the user’s home directory.
- **max-size** - The maximum number of commands stored in the history file. The default is **500**.

**resolve-parameter-values**

Whether to resolve system properties specified as command argument (or operation parameter) values before sending the operation request to the controller. The default is **false**.

**connection-timeout**

The time in milliseconds allowed to establish a connection with the controller. The default is **5000**.

**ssl**

The configuration for the keystores and truststores used for SSL.

---

**WARNING**

Red Hat recommends that SSLv2, SSLv3, and TLSv1.0 be explicitly disabled in favor of TLSv1.1 or TLSv1.2 in all affected packages.

---

- **vault** - The vault configuration. If neither **code** nor **module** is specified, the default implementation will be used. If **code** is specified but not **module**, it will look for the specified class in the Picketbox module. If **module** and **code** are specified, it will look for the class specified by code in the module specified by 'module'.
- **key-store** - The keystore.
- **key-store-password** - The keystore password.
- **alias** - The alias.
- **key-password** - The key password.
- **trust-store** - The truststore.
- **trust-store-password** - The truststore password.
- **modify-trust-store** - If set to **true**, the CLI will prompt the user when unrecognized certificates are received and allow them to be stored in the truststore. The default is **true**.

**silent**

Whether to write informational and error messages to the terminal. The default is **false**.

**access-control**
Whether the management-related commands and attributes should be filtered for the current user based on the permissions the user has been granted. For example, if true, tab completion will hide commands and attributes that the user is not allowed to access. The default is true.

echo-command
Whether to include the prompt and command in the output for commands executed in non-interactive mode. The default is false.

command-timeout
The maximum time, in seconds, to wait for a command to complete. A value of 0 means no timeout. By default, there is no timeout.

output-json
Whether to display operation responses in pure JSON format. By default, operation responses are displayed in DMR format.

color-output
Whether to print the CLI log output in color based on the log message output type. The available colors are black, blue, cyan, green, magenta, red, white, and yellow.

- enabled - Whether or not the color output is enabled. The default is true.
- error-color - Defaults to red.
- warn-color - Defaults to yellow.
- success-color - Defaults to default, which is the terminal’s default foreground color.
- required-color - Defaults to magenta.
- workflow-color - Defaults to green.
- prompt-color - Defaults to blue.

output-paging
Whether the management CLI should pause after a page of output has been displayed, which allows you to browse and search the output. If this option is set to false, the entire output is printed immediately. The default is true.

6.1. PROPERTY SUBSTITUTION

JBoss EAP supports the use of preset element and property expressions in the management CLI. These expressions will be resolved to their defined values during the execution of the command.

You can substitute expressions for the following properties:

- the operation address part of the operation request (for example, node types or names)
- operation name
- operation parameter names
- header names and values
- command names
- command argument names
By default, the management CLI performs property substitution for every line except for argument or parameter values. Argument and parameter values are resolved in the server at runtime. If you require property substitution for argument or parameter values to occur in the management CLI and have it send the resolved values to the server, complete the following steps.

1. Edit the management CLI configuration file: `EAP_HOME/bin/jboss-cli.xml`.

2. Set the `resolve-parameter-values` parameter to `true` (the default is `false`).

   `<resolve-parameter-values>true</resolve-parameter-values>`

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

System property values used in management CLI commands must have already been defined in order to be resolved. You must either pass in a properties file (`--properties=path/to/file.properties`) or property value pairs (`-Dkey=value`) when starting your management CLI instance. The properties file uses a standard `KEY=VALUE` syntax.

Property keys are denoted in your management CLI commands using the `${MY_VAR}` syntax, for example:

`/host=${hostname}/server-config=${servername}:add(group=main-server-group)`

See Configuring the Management CLI for other `jboss-cli.xml` configuration options.

6.2. CREATING ALIASES

You can define aliases for the CLI commands and operations during a CLI session using the `alias` command.

The following example creates a new CLI command alias named `read_undertow` to read the resources in the `undertow` subsystem using the `alias` command:

`alias read_undertow='subsystem=undertow:read-resource'`

**NOTE**

Alias names can only contain alphanumeric characters and underscores.

To test the creation of `read_undertow` alias, type the alias name in the management CLI:

`read_undertow`

The result will be:

```json
{
    "outcome" => "success",
    "result" => {
        "default-security-domain" => "other",
        "default-server" => "default-server",
    }
}
To view a list of all available aliases, use the **alias** command:

```
alias
```

The result will be:

```
alias read_undertow='/subsystem=undertow:read-resource'
```

To remove an alias, use the **unalias** command:

```
unalias read_undertow
```

**NOTE**

The aliases are stored in the `.aesh_aliases` file within the user’s home folder.

### 6.3. .JBOSSCLIRC CONFIGURATION FILE

JBoss EAP contains the runtime configuration `.jbossclirc` file, which helps you to initialize the environment when a new session is launched. This file is located in `EAP_HOME/bin/` directory. The example provided in the file can be used as a template for user-specific environment setup. The `.jbossclirc` file is ideal for storing global CLI variables.

The content of the `.jbossclirc` file is a list of CLI supported commands and operations. This file is executed when a new management CLI session is launched but before the control is given to the user. If there are system properties specified with `--properties` argument, then the `.jbossclirc` file is executed after the properties have been set.

**Example .jbossclirc File**

```
set console=/subsystem=logging/console-handler=CONSOLE
```

**NOTE**

When using the `--connect` or `-c` argument, `.jbossclirc` is executed before the client is actually connected to the server.

The following locations will be checked for the presence of the `.jbossclirc` file in the following order:
1. If the system property `jboss.cli.rc` is defined, its value will be considered a path to the file.

2. User’s working directory as defined by the `user.dir` system property.

3. The `EAP_HOME/bin` directory.

### 6.4. USING VARIABLES

**Using the Set Command**

You can define a certain path of the server model to a variable using the `set` command. For example:

```
set s1=/host=master/server=server-one
```

This is useful in a managed domain as you can include references to host and profiles using variables to easily replicate scripts on different servers. For example:

```
$s1/subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
```

**NOTE**

The variables are referenced using `$`.

**Using the Unset Command**

You can remove the variable by using the `unset` command:

```
unset prod_db
```

**Using the jbossclirc File**

To use the variable across CLI session, you can include these variables in the `.jbossclirc` file. This file is located in the `EAP_HOME/bin/` directory.

For example:

```
set s1=/host=master/server=server-one
set s2=/host=master/server=server-two
```

Now, restart the management CLI and issue a `set` command to check the available variables:

```
set
```

The output will be:

```
s1=/host=master/server=server-one
s2=/host=master/server=server-two
```

The variables may appear in any part of a command line and resolved during the command line parsing phase. In this example, the `prod_db` variable will be resolved to a datasource:

```
$prod_db/statistics=jdbc:read-resource
```

**Using the Echo Command**
Use the `echo` command to check the value of a variable:

```
$ echo $prod_db
```

The output will be:

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

**Example**

The following general examples show where the variables may appear and that the entire command line may consist of variables:

```
$prod_db:$op($param=$param_value)
$cmd --$param=$param_value
```

**NOTE**

The variables help you in CLI scripting.
CHAPTER 7. MANAGEMENT CLI COMMAND HISTORY

The management CLI features a command history functionality that is enabled by default in the application server installation. The history is kept both as a record in the volatile memory of the active CLI session, and appended to a log file that saves automatically in the user’s home directory as .jboss-cli-history. This history file is configured by default to record up to a maximum of 500 CLI commands. The history file location and maximum history entries can be customized in the EAP_HOME/bin/jboss-cli.xml file.

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

View the Management CLI Command History
Display the CLI command history stored in memory since the management CLI startup or the history clear command.

```
history
```

Clear the Management CLI Command History
Clear the history of CLI commands from the session memory and from the .jboss-cli-history file saved to the user’s home directory.

```
history --clear
```

Enable the Management CLI Command History
Record CLI commands in the session memory and in the .jboss-cli-history file saved to the user’s home directory.

```
history --enable
```

Disable the Management CLI Command History
Do not record CLI commands in the session memory or in the .jboss-cli-history file saved to the user’s home directory.

```
history --disable
```
CHAPTER 8. MANAGEMENT CLI LOGGING

You can capture output and other management CLI information in a log file. By default, management CLI logging is disabled. You can enable it and configure other logging settings using the EAP_HOME/bin/jboss-cli-logging.properties file.

Configure Management CLI Logging

1. Edit the EAP_HOME/bin/jboss-cli-logging.properties file.

2. Uncomment or add the following line to enable logging.

   # uncomment to enable logging to the file
   logger.handlers=FILE

3. Change the log level from OFF to the desired level, such as INFO or ALL.

   logger.org.jboss.as.cli.level=INFO

Once you restart the management CLI, output will be logged to the EAP_HOME/bin/jboss-cli.log file.

For information on configuring other settings in a logging properties file, see the Configuring logging.properties section of the JBoss EAP Development Guide.
Batch processing allows multiple operation requests to be grouped in a sequence and executed together as a unit. If any of the operation requests in the sequence fail, the entire group of operations is rolled back.

**NOTE**

Batch mode does not support conditional statements.

1. Enter batch mode with the `batch` management CLI command.

   ```
   batch
   ```

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

2. Add operation requests to the batch.

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

   You can edit and reorder batch commands. You can also store a batch for processing at a later time. See [Batch Mode Commands](#) for a full list of commands available for working with batches.

3. Run the batch.

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

   ```
   run-batch
   ```

   The entered sequence of operation requests is completed as a batch and prints the result to the terminal: The batch executed successfully.

**Batch Commands in External Files**

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

You can create a batch command file by using a text editor and placing each command on its own line.

The following command will load the `myscript.txt` file in batch mode. The commands from this file can then be edited or removed. New commands can be inserted. Changes made in this batch session do not persist to the `myscript.txt` file.

```
batch --file=myscript.txt
```

The following will immediately run the batch commands stored in the file `myscript.txt`

```
run-batch --file=myscript.txt
```

The entered sequence of operation requests is completed as a batch.
CHAPTER 10. EMBEDDING A SERVER FOR OFFLINE CONFIGURATION

You can embed a JBoss EAP standalone server or host controller process inside the management CLI process. This allows you to configure the server without it being visible on the network. A common use of this feature is for initial configuration of the server, such as managing security-related settings or avoiding port conflicts, prior to the server being online.

This direct, local administration of a JBoss EAP installation through the management CLI does not require a socket-based connection. You can use the management CLI with the embedded server in a way that is consistent with interacting with a remote JBoss EAP server. All of the standard management CLI commands that you can use to administer a remote server are available.

Start an Embedded Standalone Server

You can launch a standalone server locally using the management CLI to modify standalone configuration without launching an additional process or opening network sockets.

The following procedure launches the management CLI, starts an embedded standalone server, modifies configuration, and then stops the embedded server.

1. Launch the management CLI.
   
   $ EAP_HOME/bin/jboss-cli.sh

2. Launch the embedded standalone server.
   Passing in the --std-out=echo parameter prints the standard output to the terminal.

   embed-server --std-out=echo

3. Perform the desired operations.
   
   /socket-binding-group=standard-sockets/socket-binding=management-http:write-attribute(name=port,value=9991)

4. Stop the embedded server.

   stop-embedded-server

   This stops the embedded server and returns you to your management CLI session. If you want to exit the management CLI session as well, you can use the quit command.

Specifying the Server Configuration

By default, the embedded server will use the standalone.xml configuration file. You can use the --server-config parameter to specify a different configuration file to use.

   embed-server --server-config=standalone-full-ha.xml

Starting in Admin-only Mode

By default, the embedded server is started in admin-only mode, which will start services related to server administration, but will not start other services or accept end-user requests. This is useful for the initial configuration of the server.
You can start the embedded server in the normal running mode by setting the `--admin-only` parameter to false.

```
embed-server --admin-only=false
```

You can also change the running mode using the `reload` command.

```
reload --start-mode=normal
```

**Controlling Standard Output**

You can control how to handle standard output from the embedded server. By default, standard output is discarded, but you could find the output in the server log. You can pass in `--std-out=echo` to have server output appear with the management CLI output.

```
embed-server --std-out=echo
```

**Boot Timeout**

By default, the `embed-server` command blocks indefinitely waiting for the embedded server to fully start. You can specify the time to wait in seconds using the `--timeout` parameter. A value less than 1 will return as soon as the embedded server reaches a point where it can be managed by the CLI.

```
embed-server --timeout=30
```

**Starting with a Blank Configuration**

When starting an embedded server, you can specify to start with an empty configuration. This is useful if you want to build the entire server configuration using management CLI commands.

```
embed-server --server-config=my-config.xml --empty-config
```

This command will fail if the file already exists, which helps to avoid the accidental deletion of a configuration file. You can specify to remove any existing configuration by passing in the `--remove-existing` parameter.

```
embed-server --server-config=my-config.xml --empty-config --remove-existing
```

**Start an Embedded Host Controller**

You can launch a host controller locally using the management CLI to modify domain and host controller configuration without launching additional processes or opening network sockets.

An embedded host controller does not start any of its servers. Additionally, you can not use the `--admin-only` parameter when starting an embedded host controller. It will always be launched as if it is in `admin-only` mode.

The following procedure launches the management CLI, starts an embedded host controller, modifies configuration, and then stops the embedded host controller.

1. Launch the management CLI.
   ```
   $ EAP_HOME/bin/jboss-cli.sh
   ```
2. Launch the embedded host controller.
   ```
   Passing in the `--std-out=echo` parameter prints the standard output to the terminal.
   ```
3. Perform the desired operations.

4. Stop the embedded host controller.

Specifying the Host Controller Configuration
By default, the embedded host controller will use domain.xml for domain configuration and host.xml for host configuration. You can use the --domain-config and --host-config parameters to specify different configuration files to use.

```bash
embed-host-controller --domain-config=other-domain.xml --host-config=host-slave.xml
```

NOTE
Depending on which alternative configuration file you use, you may need to set certain properties when launching the management CLI. For example,

```bash
$ EAP_HOME/bin/jboss-cli.sh -Djboss.domain.master.address=127.0.0.1
```

Controlling Standard Output
You can control how to handle standard output from the embedded host controller. By default, standard output is discarded, but you could find the output in the host controller's log. You can pass in --std-out=echo to have host controller output appear with the management CLI output.

```bash
embed-host-controller --std-out=echo
```

Boot Timeout
By default, the embed-host-controller command blocks indefinitely waiting for the embedded host controller to fully start. You can specify the time to wait in seconds using the --timeout parameter. A value less than 1 will return as soon as the embedded host controller reaches a point where it can be managed by the CLI.

```bash
embed-host-controller --timeout=30
```

Non-Modular Class Loading with the Management CLI
Using the EAP_HOME/bin/jboss-cli.sh script to launch the management CLI uses a modular class loading environment. If you use the EAP_HOME/bin/client/jboss-cli-client.jar to run the management CLI in a non-modular class loading environment, you will need to specify the root JBoss EAP installation directory.

1. Launch the management CLI.

   ```bash
   $ java -jar EAP_HOME/bin/client/jboss-cli-client.jar
   ```

2. Start the embedded server, specifying the root installation directory.
embed-server --jboss-home=/path/to/EAP_HOME

NOTE
To embed a host controller, use the embed-host-controller command.

The embedding logic will set up an appropriate modular class loading environment for the server. The module path for the modular class loader will have a single element: 
EAP_HOME/modules.

No matter which way you launch the management CLI, the embedded server will run in a modular class loading environment.
CHAPTER 11. HOW TO...

The following CLI commands and operations provide basic examples on how to accomplish certain tasks. For detailed instructions, see the appropriate section of the Configuration Guide, Configuring Messaging, or other JBoss EAP documentation.

Unless specified otherwise, the examples apply when running as a standalone server. Use the --help argument on a command to get usage for that command. Use the read-operation-description to get information on a particular operation for a resource.

11.1. ADD A DATASOURCE

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --connection-url=CONNECTION_URL
```

11.2. ADD AN EXTENSION

Example: Add a New Extension to a Configuration

```
/extension=EXTENSION_NAME:add
```

11.3. ADD A JMS QUEUE

```
jms-queue add --queue-address=QUEUE_NAME --entries=JNDI_NAME
```

11.4. ADD A JMS TOPIC

```
jms-topic add --topic-address=TOPIC_NAME --entries=JNDI_NAME
```

11.5. ADD A MODULE

```
module add --name=MODULE_NAME --resources=PATH_TO_RESOURCE --dependencies=DEPENDENCIES
```
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. For more information, see the Create a Custom Module Manually and Remove a Custom Module Manually sections of the JBoss EAP Configuration Guide.

Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

11.6. ADD A SERVER

Example: Add a New Server to a Host in a Managed Domain

```
/host=HOST_NAME/server-config=SERVER_NAME:add(group=SERVER_GROUP_NAME)
```

11.7. ADD A SERVER GROUP

Example: Add a New Server Group in a Managed Domain

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

11.8. ADD A SYSTEM PROPERTY

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

11.9. CLONE A PROFILE

Example: Clone a Profile in a Managed Domain

```
/profile=PROFILE_TO_CLONE:clone(to-profile=NEW_PROFILE_NAME)
```

11.10. CREATE A HIERARCHICAL PROFILE

Example: Create a New Profile That Inherits from Other Profiles

```
/profile=NEW_PROFILE_NAME:add(includes=[PROFILE_1,PROFILE_2])
```

11.11. DEPLOY AN APPLICATION TO A MANAGED DOMAIN
Example: Deploy an Application to All Server Groups

```
deployment deploy-file /path/to/DEPLOYMENT.war --all-server-groups
```

Example: Deploy an Application to One or More Server Groups

```
deployment deploy-file /path/to/DEPLOYMENT.war --server-groups=SERVER_GROUP_1,SERVER_GROUP_2
```

11.12. DEPLOY AN APPLICATION TO A STANDALONE SERVER

```
deployment deploy-file /path/to/DEPLOYMENT.war
```

11.13. DISABLE ALL APPLICATIONS

```
deployment disable /path/to/DEPLOYMENT.war
```

You can use the deployment disable-all command to disable all the deployments.

```
deployment disable-all
```

11.14. DISPLAY THE ACTIVE USER

Example: Command to Display the Current User

```
:whoami
```

Example: Output for the Current User

```
{
  "outcome" => "success",
  "result" => {
    "identity" => {
      "username" => "$local",
      "realm" => "ManagementRealm"
    }
  }
}
```

11.15. DISPLAY THE CONTENTS OF AN ATTACHMENT

You can use the attachment display command to display the contents of an attachment returned from a management operation. This applies to any management operation that returns the attached-streams response header.

For example, the following operation returns the server.log file attached as a stream.

```
/subsystem=logging/log-file=server.log:read-attribute(name=stream)
{
  "outcome" => "success",
  "result" => "f61a27c4-c5a7-43ac-af1f-29e90c9acb3e",
}
You can use the `attachment display` command to display the contents of the stream returned from this operation to the console.

```
attachment display --operation=/subsystem=logging/log-file=server.log:read-attribute(name=stream)
```

This outputs the contents of the `server.log` file to the console.

```
ATTACHMENT 3480a327-31dd-4412-bdf3-f36c94ac4a09:
2019-10-18 09:19:37,366 INFO  [org.jboss.msc] (main) JBoss MSC version 1.4.5.Final-redhat-00001
2019-10-18 09:19:37,380 INFO  [org.jboss.threads] (main) JBoss Threads version 2.3.2.Final-redhat-1
2019-10-18 09:19:37,510 INFO  [org.jboss.as] (MSC service thread 1-1) WFLYSRV0049: JBoss EAP 7.3.0.GA (WildFly Core 10.0.0.Final-redhat-20190924) starting
...
```

### 11.16. DISPLAY SCHEMA INFORMATION

To show the schema information for the `:product-info` command:

```
:read-operation-description(name=product-info)
```

To display the schema version, execute an `ls` command at the management CLI root and look for the `management-*:version` values:

```
...  
management-major-version=4  
management-micro-version=0  
management-minor-version=1  
...
```

### 11.17. DISPLAY SYSTEM AND SERVER INFORMATION

Example: Command to Display the System and Server Information

```
:product-info
```

Example: Output for the System and Server Information

```
{  
  "outcome" => "success",  
  "result" => {  
    "summary" => {  
      "host-name" => "HOST_NAME",  
      "instance-identifier" => "INSTANCE_ID",  
      "product-name" => "JBoss EAP",  
    }  
  }  
}```
"product-version" => "7.3.0.GA",
"product-community-identifier" => "Product",
"product-home" => "EAP_HOME",
"standalone-or-domain-identifier" => "OPERATING_MODE",
"host-operating-system" => "OS_NAME",
"host-cpu" => {
    "host-cpu-arch" => "CPU_ARCH",
    "host-core-count" => "CORE_COUNT"
},
"jvm" => {
    "name" => "JAVA_VM_NAME",
    "java-version" => "JAVA_VERSION",
    "jvm-version" => "JAVA_VM_VERSION",
    "jvm-vendor" => "JAVA_VM_VENDOR",
    "java-home" => "JAVA_HOME"
}
}
]]

Similarly, for a managed domain, you can display the information for a particular JBoss EAP host or server:

```
/host=HOST_NAME:product-info
```

```
/host=HOST_NAME/server=SERVER_NAME:product-info
```

11.18. ENABLE ALL DISABLED DEPLOYMENTS

```
deployment enable DEPLOYMENT.war
```

You can use the `deployment enable-all` command to enable all the deployments.

```
deployment enable-all --server-groups=other-server-group
```

11.19. GET THE COMMAND TIMEOUT VALUE

Example: Display the CLI Command Timeout Value

```
command-timeout get
```

The value returned is in seconds. A value of 0 means no timeout.

11.20. RELOAD A HOST CONTROLLER

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

11.21. RELOAD A HOST CONTROLLER IN ADMIN-ONLY MODE

```
reload --host=HOST_NAME --admin-only=true
```
11.22. RELOAD ALL SERVERS IN A SERVER GROUP

Example: Reload All Servers in a Certain Server Group in a Managed Domain

```
/server-group=SERVER_GROUP_NAME:reload-servers
```

**NOTE**

To reload the servers in a suspended state, pass in the `start-mode=suspend` argument.

11.23. RELOAD A SERVER

Example: Reload a Server in a Managed Domain

```
/host=HOST_NAME/server-config=SERVER_NAME:reload
```

**NOTE**

To reload the server in a suspended state, pass in the `start-mode=suspend` argument.

11.24. RELOAD A STANDALONE SERVER

```
reload
```

**NOTE**

To reload the server in admin-only mode, pass in the `--start-mode=admin-only` argument. To reload the server in a suspended state, pass in the `--start-mode=suspend` argument.

11.25. REMOVE AN EXTENSION

Example: Remove an Existing Extension

```
/extension=EXTENSION_NAME:remove
```

11.26. REMOVE A MODULE

```
module remove --name=MODULE_NAME
```
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. For more information, see the Create a Custom Module Manually and Remove a Custom Module Manually sections of the JBoss EAP Configuration Guide.

Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See Technology Preview Features Support Scope on the Red Hat Customer Portal for information about the support scope for Technology Preview features.

11.27. RESET THE COMMAND TIMEOUT VALUE

**Example: Reset the Command Timeout to the Default Value**

```
command-timeout reset default
```

**Example: Reset the Command Timeout to the Value Provided by the CLI Configuration**

```
command-timeout reset config
```

**NOTE**

The value provided by the CLI configuration can be set in either in the `EAP_HOME/bin/jboss-cli.xml` file or passed in with the `--command-timeout` argument when starting the management CLI.

11.28. RESTART ALL SERVERS IN A SERVER GROUP

**Example: Restart All Servers in a Certain Server Group in a Managed Domain**

```
/server-group=SERVER_GROUP_NAME:restart-servers
```

**NOTE**

To restart the servers in a suspended state, pass in the `start-mode=suspend` argument.

11.29. RESTART A SERVER

**Example: Restart a Server in a Managed Domain**

```
/host=HOST_NAME/server-config=SERVER_NAME:restart
```
NOTE
To restart the server in a suspended state, pass in the `start-mode=suspend` argument.

11.30. SAVE THE CONTENTS OF AN ATTACHMENT

You can use the `attachment save` command to save the contents of an attachment returned from a management operation to a file. This applies to any management operation that returns the `attached-streams` response header.

For example, the following operation returns the `server.log` file attached as a stream.

```
/subsystem=logging/log-file=server.log:read-attribute(name=stream)
{
    "outcome" => "success",
    "result" => "f61a27c4-c5a7-43ac-af1f-29e90c9acb3e",
    "response-headers" => {"attached-streams" => [{
        "uuid" => "f61a27c4-c5a7-43ac-af1f-29e90c9acb3e",
        "mime-type" => "text/plain"
    }]}
}
```

You can use the `attachment save` command to save the contents of the stream returned form this operation to a file.

```
attachment save --operation="/subsystem=logging/log-file=server.log:read-attribute(name=stream)" -- file=log-output.txt
```

This saves the contents of the `server.log` file to `EAP_HOME/bin/log-output.txt`.

11.31. SET THE COMMAND TIMEOUT VALUE

Example: Set the Maximum Time to Wait for a CLI Command to Complete

```
command-timeout set TIMEOUT_VALUE
```

The value is set in seconds. A value of 0 means no timeout.

11.32. SHUT DOWN A HOST CONTROLLER

Example: Shut Down a Host Controller in a Managed Domain

```
shutdown --host=HOST_NAME
```

11.33. SHUT DOWN THE SERVER

Example: Shut Down a Standalone Server

```
shutdown
```
11.34. START ALL SERVERS IN A SERVER GROUP

Example: Start All Servers in a Certain Server Group in a Managed Domain

/server-group=SERVER_GROUP_NAME:start-servers

**NOTE**
To start the servers in a suspended state, pass in the `start-mode=suspend` argument.

11.35. START A SERVER

Example: Start a Server in a Managed Domain

/host=HOST_NAME/server-config=SERVER_NAME:start

**NOTE**
To start the server in a suspended state, pass in the `start-mode=suspend` argument.

11.36. STOP ALL SERVERS IN A SERVER GROUP

Example: Stop All Servers in a Certain Server Group in a Managed Domain

/server-group=SERVER_GROUP_NAME:stop-servers

11.37. STOP A SERVER

Example: Stop a Server in a Managed Domain

/host=HOST_NAME/server-config=SERVER_NAME:stop

11.38. TAKE A CONFIGURATION SNAPSHOT

Example: Take a Snapshot of the Current Configurations

:take-snapshot

11.39. UNDEPLOY ALL APPLICATIONS

Example: Undeploy All Applications from a Managed Domain

deployment undeploy * --all-relevant-server-groups

Example: Undeploy All Applications from a Standalone Domain

deployment undeploy *
11.40. UNDEPLOY AN APPLICATION FROM A MANAGED DOMAIN

Example: Undeploy an Application from All Server Groups with That Deployment

```
deployment undeploy DEPLOYMENT.war --all-relevant-server-groups
```

Example: Undeploy an Application from a Specific Server Group

```
deployment undeploy DEPLOYMENT.war --server-groups=SERVER_GROUP_NAME
```

11.41. UNDEPLOY AN APPLICATION FROM A STANDALONE SERVER

```
deployment undeploy DEPLOYMENT.war
```

11.42. UPDATE A HOST NAME

Example: Update the Name of a Host in a Managed Domain

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

The host must be reloaded in order for the changes to take effect.

11.43. UPLOAD AN ATTACHMENT

You can upload a local file as an attachment to management operations that accept file streams. For example, the following management CLI command uses the `input-stream-index` option to upload the contents of a local file to an exploded deployment.

```
/deployment=DEPLOYMENT_NAME.war:add-content(content=[{target-path=/path/to/FILE_IN_DEPLOYMENT, input-stream-index=/path/to/LOCAL_FILE_TO_UPLOAD}]
```

For more details on uploading files to a deployment, see the Add Content to an Exploded Deployment section of the Configuration Guide.

11.44. VIEW A SERVER LOG

```
/subsystem=logging/log-file=SERVER_LOG_NAME:read-log-file
```
A.1. MANAGEMENT CLI STARTUP ARGUMENTS

The following table lists the arguments that can be passed into the `jboss-cli` script to launch the management CLI.

Table A.1. Management CLI Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--bind</code></td>
<td>Specifies to which address the CLI is going to be bound to. If none is provided then the CLI will choose one automatically as needed.</td>
</tr>
<tr>
<td><code>--command</code></td>
<td>Specifies a single command or an operation that should be executed in the CLI session. The CLI will terminate the session immediately after the command or the operation has been executed.</td>
</tr>
<tr>
<td><code>--command-timeout</code></td>
<td>The maximum time, in seconds, to wait for a command to complete. A value of 0 means no timeout. By default, there is no timeout.</td>
</tr>
<tr>
<td><code>--commands</code></td>
<td>Specifies a comma-separated list (must not contain whitespace) of commands and operations that should be executed in the CLI session. The CLI session will terminate after the last command has been executed or if a command fails.</td>
</tr>
<tr>
<td><code>--connect, -c</code></td>
<td>Instructs the CLI to connect to the controller on start-up. This avoid having to issue a separate <code>connect</code> command later.</td>
</tr>
<tr>
<td><code>--controller</code></td>
<td>The default controller host, port, and protocol to connect to when the <code>--connect</code> option is specified upon startup or when the <code>connect</code> command is issued in the management CLI without arguments. The default host is <code>localhost</code>, the default port is 9990, and the default protocol is <code>remote+http</code>. Alternatively, a controller alias containing the host, port, and protocol information can be supplied.</td>
</tr>
<tr>
<td><code>--echo-command</code></td>
<td>Include the prompt and command in the output for commands executed in non-interactive mode.</td>
</tr>
<tr>
<td><code>--error-on-interact</code></td>
<td>Disables prompts for security-related input in non-interactive mode. If an input is required to proceed, the CLI process will terminate abruptly with an error.</td>
</tr>
<tr>
<td><code>--file</code></td>
<td>Specifies the path to a file which contains commands and operations (one per line) that should be executed non-interactively. The CLI will terminate after the last command has been executed or if a command or operation fails.</td>
</tr>
</tbody>
</table>
### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--gui</td>
<td>Launches a GUI that is built on top of the command-line interface. This GUI is not a supported way to interact with the JBoss EAP management CLI and should only be used to view configuration and help build CLI commands.</td>
</tr>
<tr>
<td>--help, -h</td>
<td>Displays the help message.</td>
</tr>
<tr>
<td>--no-color-output</td>
<td>Disable CLI output and prompt colors.</td>
</tr>
<tr>
<td>--no-local-auth</td>
<td>Disables the local authentication mechanism which allows the CLI to demonstrate that it is being executed locally to the server being managed through an exchange of tokens using the file system.</td>
</tr>
<tr>
<td>--no-output-paging</td>
<td>Disables output paging, which is when the management CLI pauses after a page of output has been displayed, allowing you to browse and search the output. When this option is specified, the entire output is printed immediately.</td>
</tr>
<tr>
<td>--output-json</td>
<td>Displays operation responses in pure JSON format. By default, operation responses are displayed in DMR format.</td>
</tr>
<tr>
<td>--password, -p</td>
<td>Specifies the password for authentication while connecting to the controller. If the argument is not specified and authentication is required, then the user will be prompted to enter the password when the <code>connect</code> command is issued.</td>
</tr>
<tr>
<td>--properties</td>
<td>Specifies a path to a properties file that contains property value pairs to define system properties. The properties file uses a standard <code>KEY=VALUE</code> syntax.</td>
</tr>
<tr>
<td>--resolve-parameter-values</td>
<td>Resolve system properties specified as command argument values or operation parameter values before sending the operation request to the controller.</td>
</tr>
<tr>
<td>--timeout</td>
<td>Specifies the time, in milliseconds, to wait for a connection to succeed. The default is 5000.</td>
</tr>
<tr>
<td>--user, -u</td>
<td>Specifies the user name if the controller requires user authentication. If the argument is not specified and authentication is required, then the user will be prompted to enter the user name when the <code>connect</code> command is issued. Local authentication is automatically disabled if a user is specified.</td>
</tr>
<tr>
<td>--version</td>
<td>Displays the application server version and environment information.</td>
</tr>
</tbody>
</table>

### A.2. MANAGEMENT CLI BATCH MODE COMMANDS

This table provides a list of commands that can be used with the management CLI to work with batches.
Table A.2. Management CLI Batch Mode Commands

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear-batch</td>
<td>Removes all the existing command lines from the currently active batch.</td>
</tr>
<tr>
<td>discard-batch</td>
<td>Discards the currently active batch and exits batch mode.</td>
</tr>
<tr>
<td>edit-batch-line</td>
<td>Edit a line in the current batch by providing the line number to edit and</td>
</tr>
<tr>
<td></td>
<td>the edited command. For example: <code>edit-batch-line 2 data-source disable --name=ExampleDS</code>.</td>
</tr>
<tr>
<td>holdback-batch</td>
<td>Postpone or store a current batch. Using this command without</td>
</tr>
<tr>
<td></td>
<td>arguments creates an unnamed heldback batch. To return to this</td>
</tr>
<tr>
<td></td>
<td>heldback batch, simply type <code>batch</code> again at the CLI command line.</td>
</tr>
<tr>
<td></td>
<td>There can be only one unnamed heldback batch.</td>
</tr>
<tr>
<td></td>
<td>You can optionally provide a name under which to store the batch by</td>
</tr>
<tr>
<td></td>
<td>using the <code>holdback_name</code> argument. To return to the named batch, pass the</td>
</tr>
<tr>
<td></td>
<td><code>holdback_name</code> to the <code>batch</code> command.</td>
</tr>
<tr>
<td></td>
<td>Use the <code>batch -l</code> command to see a list of all heldback batches.</td>
</tr>
<tr>
<td>list-batch</td>
<td>Lists all commands in the currently active batch.</td>
</tr>
<tr>
<td>move-batch-line</td>
<td>Re-order the lines in the batch by specifying the line number you want</td>
</tr>
<tr>
<td></td>
<td>to move as the first argument and its new position as the second</td>
</tr>
<tr>
<td></td>
<td>argument. For example: <code>move-batch-line 3 1</code>.</td>
</tr>
<tr>
<td>remove-batch-line</td>
<td>Removes the batch command at the specified line. For example:</td>
</tr>
<tr>
<td></td>
<td><code>remove-batch-line 3</code>.</td>
</tr>
<tr>
<td>run-batch</td>
<td>Runs the currently active batch. If the batch executed successfully, the</td>
</tr>
<tr>
<td></td>
<td>batch will be discarded and the CLI will exit batch mode.</td>
</tr>
</tbody>
</table>

A.3. MANAGEMENT CLI COMMANDS

The following table lists management CLI commands and their purposes. For more usage and argument details, use the `--help` argument on a specific command.

Table A.3. Management CLI Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>Define an alias with the format <code>NAME=VALUE</code>. When no arguments are specified,</td>
</tr>
<tr>
<td></td>
<td>the list of aliases are displayed.</td>
</tr>
<tr>
<td>batch</td>
<td>Starts batch mode by creating a new batch. If there is an unnamed heldback</td>
</tr>
<tr>
<td></td>
<td>batch, it will be reactivated. If there are named held back batches,</td>
</tr>
<tr>
<td></td>
<td>reactivate by specifying the <code>holdback_name</code>.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cd</td>
<td>Changes to the specified path.</td>
</tr>
<tr>
<td>clear</td>
<td>Clears the screen.</td>
</tr>
<tr>
<td>command</td>
<td>Allows you to add, remove, and list existing generic type commands. A generic type command is a command that is assigned to a specific node type and which allows you to perform any operation available for an instance of that type. It can also modify any of the properties exposed by the type on any existing instance.</td>
</tr>
<tr>
<td>connect</td>
<td>Connects to the controller on the specified host and port using the specified protocol when the management CLI was launched. If not specified, the default host is <strong>localhost</strong>, the default port is <strong>9990</strong>, and the default protocol is <strong>http</strong>.</td>
</tr>
<tr>
<td>connection-factory</td>
<td>Manages connection factories in the <strong>messaging-activemq</strong> subsystem.</td>
</tr>
<tr>
<td>connection-info</td>
<td>Displays information about the current connection to the server.</td>
</tr>
<tr>
<td>data-source</td>
<td>Manages datasource configuration in the <strong>datasources</strong> subsystem.</td>
</tr>
<tr>
<td>deployment deploy-file</td>
<td>Deploys an application. Use wildcard (*) to deploy all the applications.</td>
</tr>
<tr>
<td>deployment disable</td>
<td>Disables a pre-existing deployment.</td>
</tr>
<tr>
<td>deployment enable</td>
<td>Enables a pre-existing deployment.</td>
</tr>
<tr>
<td>deployment info</td>
<td>Displays information about an individual deployment or about multiple deployments.</td>
</tr>
<tr>
<td>deployment undeploy</td>
<td>Undeploys an application with the specified name.</td>
</tr>
<tr>
<td>deployment-overlay</td>
<td>Manage deployment overlays. If no arguments are specified, all existing deployment overlays are listed.</td>
</tr>
<tr>
<td>echo</td>
<td>Outputs the specified text to the console.</td>
</tr>
<tr>
<td>echo-dmr</td>
<td>Builds a DMR request for the command or operation passed in as the argument and echos in its <strong>toString</strong>() format.</td>
</tr>
<tr>
<td>help</td>
<td>Displays the help message. Can be used with an argument to display help information for a particular command or operation. Use the <strong>commands</strong> argument to display a list of available commands.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>history</td>
<td>Displays the CLI command history in memory and displays a status of whether the history expansion is enabled or disabled. Can be used with arguments to clear, disable and enable the history as required.</td>
</tr>
<tr>
<td>if</td>
<td>Starts if-else control flow.</td>
</tr>
<tr>
<td>jdbc-driver-info</td>
<td>Displays information about the installed JDBC drivers.</td>
</tr>
<tr>
<td>jms-queue</td>
<td>Manage JMS queues in the <code>messaging-activemq</code> subsystem.</td>
</tr>
<tr>
<td>jms-topic</td>
<td>Manage JMS topics in the <code>messaging-activemq</code> subsystem.</td>
</tr>
<tr>
<td>ls</td>
<td>Lists the contents of the node path. Use the <code>-l</code> switch to print the result one per line.</td>
</tr>
<tr>
<td>module</td>
<td>Add and remove modules. Note that this command is provided as Technology Preview only.</td>
</tr>
<tr>
<td>patch</td>
<td>Apply or roll back a patch to the server.</td>
</tr>
<tr>
<td>pwd</td>
<td>Prints the full node path of the current working node.</td>
</tr>
<tr>
<td>quit</td>
<td>Terminates the command line interface.</td>
</tr>
<tr>
<td>read-attribute</td>
<td>Prints the value and, depending on the arguments, the description of the attribute of a managed resource.</td>
</tr>
<tr>
<td>read-operation</td>
<td>Displays the description of a specified operation, or lists all available operations if none is specified.</td>
</tr>
<tr>
<td>reload</td>
<td>Sends the <code>:reload</code> operation request to the server/domain controller and waits for the controller to close the connection and then it returns the control back to the client.</td>
</tr>
<tr>
<td>rollout-plan</td>
<td>Manage stored rollout plans.</td>
</tr>
<tr>
<td>run-batch</td>
<td>Runs the currently active batch while in batch mode. While not in batch mode, can use with the <code>--file</code> argument to execute the contents of the file as a batch.</td>
</tr>
<tr>
<td>set</td>
<td>Initializes variables with the given names with the specified values.</td>
</tr>
<tr>
<td>shutdown</td>
<td>Sends the <code>:shutdown</code> operation request to the server/domain controller and waits for the controller to close the connection.</td>
</tr>
<tr>
<td>try</td>
<td>Starts a try-catch-finally control flow.</td>
</tr>
</tbody>
</table>
### A.4. MANAGEMENT CLI OPERATIONS

The following table lists management CLI operations that are available at the root level (/). The actual available operations for a particular resource will vary per resource and also depend on the operating mode (standalone server or managed domain).

Operations are invoked using a colon (:). The available operations for a resource can be exposed by using the `read-operation-names` operation or by using tab completion after a colon. Operation descriptions can be displayed by using the `read-operation-description` operation. For example:

```
:read-operation-description(name=write-attribute)
```

<table>
<thead>
<tr>
<th>Operation Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add-namespace</td>
<td>Adds a namespace prefix mapping to the <code>namespaces</code> attribute’s map.</td>
</tr>
<tr>
<td>add-schema-location</td>
<td>Adds a schema location mapping to the <code>schema-locations</code> attribute’s map.</td>
</tr>
<tr>
<td>clean-obsolete-content</td>
<td>Clean contents that are no longer referenced from the content repository.</td>
</tr>
<tr>
<td>delete-snapshot</td>
<td>Deletes a snapshot of the server configuration from the snapshots directory.</td>
</tr>
<tr>
<td>full-replace-deployment</td>
<td>Adds previously uploaded deployment content to the list of content available for use, replace existing content of the same name in the runtime, and remove the replaced content from the list of content available for use.</td>
</tr>
<tr>
<td>list-add</td>
<td>Add an entry to a list attribute.</td>
</tr>
<tr>
<td>list-clear</td>
<td>Clear all entries from a list attribute.</td>
</tr>
<tr>
<td>list-get</td>
<td>Get an entry from a list attribute.</td>
</tr>
<tr>
<td>Operation Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>list-remove</td>
<td>Remove an entry from a list attribute.</td>
</tr>
<tr>
<td>list-snapshots</td>
<td>Lists the snapshots of server configurations saved in the snapshots directory.</td>
</tr>
<tr>
<td>map-clear</td>
<td>Clear all entries from a map attribute.</td>
</tr>
<tr>
<td>map-get</td>
<td>Get an entry from a map attribute.</td>
</tr>
<tr>
<td>map-put</td>
<td>Add an entry to a map attribute.</td>
</tr>
<tr>
<td>map-remove</td>
<td>Remove an entry from a map attribute.</td>
</tr>
<tr>
<td>product-info</td>
<td>Returns a summary of the current server installation.</td>
</tr>
<tr>
<td>query</td>
<td>Query a resource.</td>
</tr>
<tr>
<td>read-attribute</td>
<td>Displays the value of an attribute for the selected resource.</td>
</tr>
<tr>
<td>read-attribute-group</td>
<td>Displays the value of attributes for the selected group.</td>
</tr>
<tr>
<td>read-attribute-group-names</td>
<td>Displays the names of all the attribute groups under the selected resource.</td>
</tr>
<tr>
<td>read-children-names</td>
<td>Displays the names of all children under the selected resource with the given type.</td>
</tr>
<tr>
<td>read-children-resources</td>
<td>Displays information about all of a resource’s children that are of a given type.</td>
</tr>
<tr>
<td>read-children-types</td>
<td>Displays the type names of all the children under the selected resource.</td>
</tr>
<tr>
<td>read-config-as-xml</td>
<td>Displays the current configuration in XML format.</td>
</tr>
<tr>
<td>read-operation-description</td>
<td>Displays the details of an operation for the given resource.</td>
</tr>
<tr>
<td>read-operation-names</td>
<td>Displays the names of all available operations for the given resource.</td>
</tr>
<tr>
<td>read-resource</td>
<td>Displays a resource’s attribute values along with either basic or complete information about any child resources.</td>
</tr>
<tr>
<td>read-resource-description</td>
<td>Displays the description of a resource’s attributes, types of children and operations.</td>
</tr>
<tr>
<td>reload</td>
<td>Reloads the server by shutting all services down and restarting.</td>
</tr>
<tr>
<td>Operation Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>reload-servers</td>
<td>Reloads all servers currently running in the domain.</td>
</tr>
<tr>
<td>remove-namespace</td>
<td>Removes a namespace prefix mapping from the <code>namespaces</code> attribute map.</td>
</tr>
<tr>
<td>remove-schema-location</td>
<td>Removes a schema location mapping from the <code>schema-locations</code> attribute map.</td>
</tr>
<tr>
<td>replace-deployment</td>
<td>Replace existing content in the runtime with new content. The new content must have been previously uploaded to the deployment content repository.</td>
</tr>
<tr>
<td>resolve-expression</td>
<td>Accepts an expression as input (or a string that can be parsed into an expression), and resolves it against the local system properties and environment variables.</td>
</tr>
<tr>
<td>resolve-expression-on-domain</td>
<td>Accepts an expression as input (or a string that can be parsed into an expression) and resolves it against the local system properties and environment variables on all servers in the domain.</td>
</tr>
<tr>
<td>resolve-internet-address</td>
<td>Takes a set of interface resolution criteria and finds an IP address on the local machine that matches the criteria, or fails if no matching IP address can be found.</td>
</tr>
<tr>
<td>restart-servers</td>
<td>Restarts all servers currently running in the domain.</td>
</tr>
<tr>
<td>resume</td>
<td>Resumes normal operations in a suspended server.</td>
</tr>
<tr>
<td>resume-servers</td>
<td>Resumes processing on all servers in the domain.</td>
</tr>
<tr>
<td>shutdown</td>
<td>Shuts down the server with a call to <code>System.exit(0)</code>.</td>
</tr>
<tr>
<td>start-servers</td>
<td>Starts all configured servers in the managed domain that are not currently running.</td>
</tr>
<tr>
<td>stop-servers</td>
<td>Stops all servers currently running in the managed domain.</td>
</tr>
<tr>
<td>suspend</td>
<td>Suspends server operations gracefully. All current requests will complete normally, however no new requests will be accepted.</td>
</tr>
<tr>
<td>suspend-servers</td>
<td>Suspends all servers in the domain. All current operations will finish and no new operations will be allowed.</td>
</tr>
<tr>
<td>take-snapshot</td>
<td>Takes a snapshot of the server configuration and saves it to the snapshots directory.</td>
</tr>
<tr>
<td>Operation Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>undefine-attribute</td>
<td>Sets the value of an attribute of the selected resource to <code>undefined</code>.</td>
</tr>
<tr>
<td>upload-deployment-bytes</td>
<td>Indicates that the deployment content in the included byte array should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.</td>
</tr>
<tr>
<td>upload-deployment-stream</td>
<td>Indicates that the deployment content available at the included input stream index should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.</td>
</tr>
<tr>
<td>upload-deployment-url</td>
<td>Indicates that the deployment content available at the included URL should be added to the deployment content repository. Note that this operation does not indicate the content should be deployed into the runtime.</td>
</tr>
<tr>
<td>validate-address</td>
<td>Checks whether a resource with the specified address exists.</td>
</tr>
<tr>
<td>validate-operation</td>
<td>Validates that an operation is valid according to its description. Any errors present will be shown in the operation’s <code>failure-description</code>.</td>
</tr>
<tr>
<td>whoami</td>
<td>Returns the identity of the currently authenticated user.</td>
</tr>
<tr>
<td>write-attribute</td>
<td>Sets the value of an attribute for the selected resource.</td>
</tr>
</tbody>
</table>

### A.5. RESOURCE ATTRIBUTE DETAILS

The `read-resource-description` operation displays a resource’s attributes and the details about the attribute. The following table lists the possible fields that might be returned, depending on whether it is relevant for the attribute.

#### Table A.5. Resource Attribute Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-type</td>
<td>Whether an attribute can only be read, can be read and written, or is a metric. Valid values are <code>read-only</code>, <code>read-write</code>, and <code>metric</code>. A metric is a read-only attribute, where the value is not stored in the persistent configuration and can change due to activity on the server.</td>
</tr>
<tr>
<td>allowed</td>
<td>The list of valid values.</td>
</tr>
<tr>
<td>alternatives</td>
<td>Defines an exclusive relationship between attributes. If this attribute’s value is set, the attributes listed in the <code>alternatives</code> field should be undefined, even if those attributes state that they are required.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>capability-reference</td>
<td>Indicates that this attribute’s value specifies the dynamic portion of the name of the specified capability provided by another resource. This indicates the attribute is a reference to another area of the management model.</td>
</tr>
<tr>
<td>default</td>
<td>The default value to use for the attribute if a value is not provided.</td>
</tr>
<tr>
<td>description</td>
<td>The text description of the attribute.</td>
</tr>
<tr>
<td>deprecated</td>
<td>Whether this attribute is deprecated. It also provides the version that it was deprecated in and reason for the deprecation.</td>
</tr>
<tr>
<td>expressions-allowed</td>
<td>Whether the attribute’s value can be an expression.</td>
</tr>
<tr>
<td>max</td>
<td>The maximum value of a numeric attribute.</td>
</tr>
<tr>
<td>max-length</td>
<td>The maximum length of an attribute of type <code>STRING</code>, <code>LIST</code>, or <code>BYTES</code>.</td>
</tr>
<tr>
<td>min</td>
<td>The minimum value of a numeric attribute.</td>
</tr>
<tr>
<td>min-length</td>
<td>The minimum length of an attribute of type <code>STRING</code>, <code>LIST</code>, or <code>BYTES</code>.</td>
</tr>
<tr>
<td>nillable</td>
<td>Whether it is allowed for the attribute to not have a defined value. An attribute can be undefined either because it is not required or because it is required but has alternatives and an alternative is defined. This field helps users easily understand whether they need to account for the possibility of an undefined value.</td>
</tr>
<tr>
<td>required</td>
<td>Whether the attribute must have a defined value. If <code>true</code>, a value must be defined or an alternative must be defined. If <code>false</code>, the value may be undefined.</td>
</tr>
<tr>
<td>requires</td>
<td>Indicates that, if this attribute has a defined value, the attributes defined in this list must also have a value.</td>
</tr>
<tr>
<td>restart-required</td>
<td>Defines which services must be restarted upon execution of a <code>write-attribute</code> operation. This field allows the following values.</td>
</tr>
<tr>
<td></td>
<td>- <strong>no-services</strong> - No services must be restarted.</td>
</tr>
<tr>
<td></td>
<td>- <strong>all-services</strong> - All services must be restarted.</td>
</tr>
<tr>
<td></td>
<td>- <strong>resource-services</strong> - Some services associated with the resource must be restarted.</td>
</tr>
<tr>
<td></td>
<td>- <strong>jvm</strong> - The entire JVM must be restarted.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>storage</td>
<td>Whether the attribute's value is stored in the persistent configuration file or only exists as long as the resource is running. The value can be either <strong>configuration</strong> or <strong>runtime</strong>.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the attribute’s value. The allowed values are <strong>BIG_DECIMAL</strong>, <strong>BIG_INTEGER</strong>, <strong>BOOLEAN</strong>, <strong>BYTES</strong>, <strong>DOUBLE</strong>, <strong>INT</strong>, <strong>LIST</strong>, <strong>LONG</strong>, <strong>OBJECT</strong>, <strong>PROPERTY</strong>, and <strong>STRING</strong>.</td>
</tr>
<tr>
<td>value-type</td>
<td>Defines additional type information for attributes of type <strong>LIST</strong> or <strong>OBJECT</strong>. A <strong>value-type</strong> of <strong>INT</strong> for a <strong>LIST</strong> attribute is analogous to a Java <strong>List&lt;Integer&gt;</strong>. A <strong>value-type</strong> of <strong>STRING</strong> for an <strong>OBJECT</strong> attribute is analogous to a Java <strong>Map&lt;String, String&gt;</strong>. If all elements in an <strong>OBJECT</strong> attribute are not of the same type, the <strong>value-type</strong> represents a fully-defined complex object that defines that object’s fields and values.</td>
</tr>
<tr>
<td>unit</td>
<td>The unit of the attribute’s value, if appropriate.</td>
</tr>
</tbody>
</table>